Machine Learning - Análise de dados de Pokémons

Bruno Mourão e Leyberson

Professor : César Lincoln Cavalcante Mattos

Link para o projeto no github : https://github.com/brunolmourao/machine-learning

1.Introdução:

Pokémon é uma série de jogos eletrônicos desenvolvidos pela Game Freak e publicados pela Nintendo como parte da franquia de mídia Pokémon. Lançado pela primeira vez em 1996 no Japão para o console Game Boy, a principal série de jogos de RPGs, que continuou em cada geração em portáteis da Nintendo.

Os jogos são geralmente lançados em pares - cada um com pequenas variações - com uma recriação aprimorado dos usados jogos lançados em alguns anos depois das versões originais. Enquanto a série principal consiste em RPGs, os spin-off abrangem outros gêneros, como RPG de ação, quebra-cabeça e jogos virtuais para animais de estimação.

No contexto do lançamento dos próximos jogos da franquia , Pokémon Sword e Pokémon Shield , teremos novos pokémons revelados.Logo levantamos algumas perguntas : Será que é possível prever atributos desse novos pokémons baseados nos já existentes ?

Utilizando o dataset **Pokémon for Data Mining and Machine Learning** que contém os dados de todos os pokémons até a 6ª geração , incluindo algumas formas alternativas , iremos realizar o estudo em cima de 2 problemas de classificação , uma binária para classificar o pokémon como lendário ou não , outro problema de classificação multiclasse para tentarmos determinar o tipo de um pokémon baseado em seus atributos , cor , estilo do corpo e grupo do ovo.

2.Fundamentação Teórica :

Nosso dataset é constituído por 21 atributos :

•Name : O nome de uma dado Pokémon.

•Number : O número da pokédex de um dado pokémon, basicamente é uma enumeração utilizadas nos jogos para organizá-los .

• Type_1. Tipo principal do Pokémon. Relaciona-se a natureza, com seu estilo de vida e com os movimentos que é capaz de aprender ao longo do tempo de luta. Este valor é de 18 valores diferentes: Bug, Dark, Dragon, Electric, Fairy, Fighting, Fire, Flying, Ghost, Grass, Ground, Ice, Normal, Poison, Psychic, Rock, Steel e Water.

• Type_2. Pokémon pode ter dois tipos, mas nem todos. Os valores possíveis que esse tipo secundário pode receber são os mesmos que o variável Type_1.

• Total. A soma de todas as estatísticas de batalha base de um Pokémon. Deve ser um bom indicador da força geral de um Pokémon. É a soma das próximas seis variáveis. Cada um deles representa um status de batalha base. Todas as estatísticas de batalha são variáveis contínuas, porém inteiras, ou seja, o número de valores que eles podem assumir é infinito na teoria, ou apenas muito grande na prática.

• HP. Pontos de saúde básicos dos Pokémon. Quanto maior, mais tempo o Pokémon será capaz de permanecer em um combate .

• Ataque. Base de ataque do Pokémon. Quanto maior, mais dano seus ataques físicos causarão ao inimigo Pokémon.

• Defesa: Defesa base do Pokémon. Quanto maior, menos dano receberá quando for atingido por um ataque físico.

• Sp_Atk. Base de ataque especial do Pokémon. Quanto maior, mais dano seus ataques especiais causarão ao inimigo Pokémon.

• Sp_Def. Base de defesa especial do Pokémon. Quanto maior, menor o dano que receberá ao ser atingido por um ataque especial.

• Rapidez. Velocidade base do Pokémon.É o fator que determina qual pokémon ataca primeiro

• Geração. A geração em que o Pokémon foi lançado. É um número inteiro entre 1 e 6, portanto, é uma variável numérica descritiva. Pode-se verificar a análise do desenvolvimento ou crescimento do grupo nos anos anteriores

isLegendary. Booleano indicando se o Pokémon é lendário ou não. Os Pokémon lendários tendem a resistir melhor, a ter habilidades únicas , a serem extremamente difíceis de encontrar e até mesmo a capturar.

• Cor. Cor dos Pokémon de acordo com o Pokédex. O Pokédex distingue entre dez cores: preto, azul, marrom, verde, cinza, rosa, roxo, vermelho, branco e amarelo.

hasGender. Booleano indicando o Pokémon pode ser classificado como masculino ou feminino.

• Pr_Male. Caso o Pokémon tenha Gênero, a probabilidade de ele ser do sexo masculino. A probabilidade de ser feminino é, naturalmente, 1 menos este valor. Como Geração, esta variável é numérica e discreta, porque embora seja a probabilidade de o Pokémon aparecer como uma fêmea ou macho na natureza, só pode levar 7 valores: 0,0.125,0.25,0.5,0.75,0.875, e 1.

• Egg_Group_1. Valor categórico indicando o grupo de ovos dos Pokémon. Está relacionado com a raça dos Pokémon, e é um fator determinante na criação dos Pokémon.

Seus 15 valores possíveis são: Amorphous, Bug, Ditto, Dragon, Fairy, Field, Flying, Grass, Human-Like, Mineral, Monster, Undiscovered, Water_1, Water_2 e Water_3.

Egg_Group_2. Similarmente, no caso dos Pokémon, os Pokémon podem pertencer a dois grupos.

• hasMegaEvolution. Booleano indicando se um Pokémon pode mega-evoluir ou não. Mega-evolução é uma propriedade que alguns Pokémon possuem e permite que eles mudem sua aparência, tipos e atributos durante um combate em uma forma muito forte.

• Altura_m. Altura do Pokémon de acordo com o Pokédex, medido em metros. É uma variável contínua numérica.

• Weight_kg Peso do Pokémon de acordo com o Pokédex, medido quilogramas. É também uma variável contínua numérica.

• Catch_Rate. Variável numérica que indica como é fácil capturar um Pokémon ao tentar capturá-lo para torná-lo parte do seu time. Ele é limitado entre 3 e 255. O número de valores diferentes que ele leva não é muito alto, mas pode ser considerado uma variável contínua.

• Body_Style. Estilo do corpo do Pokémon de acordo com o Pokédex. São especificadas 14 categorias de estilo de corpo: bipedal_tailed, bipedal_tailless, four_wings, head_arms, head_base, head_legs, head_only, insectoide, multiple_bodies, quadruped, serpentine_body, several_limbs, two_wings e with_fins.

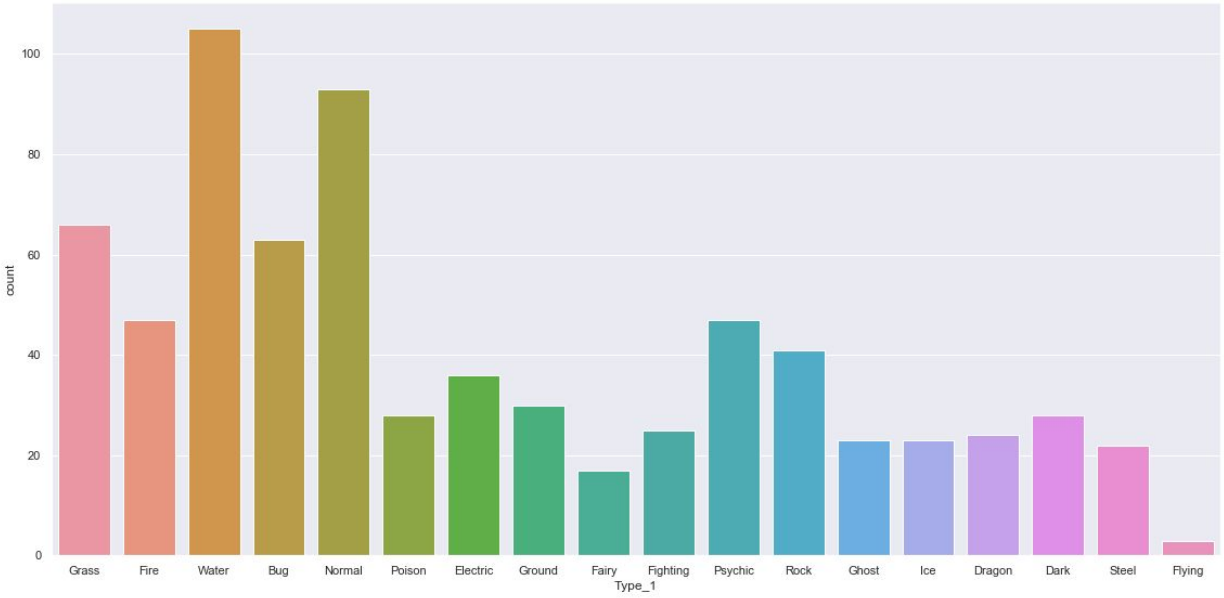


Gráfico 1: Gráfico de distribuição dos tipos primários

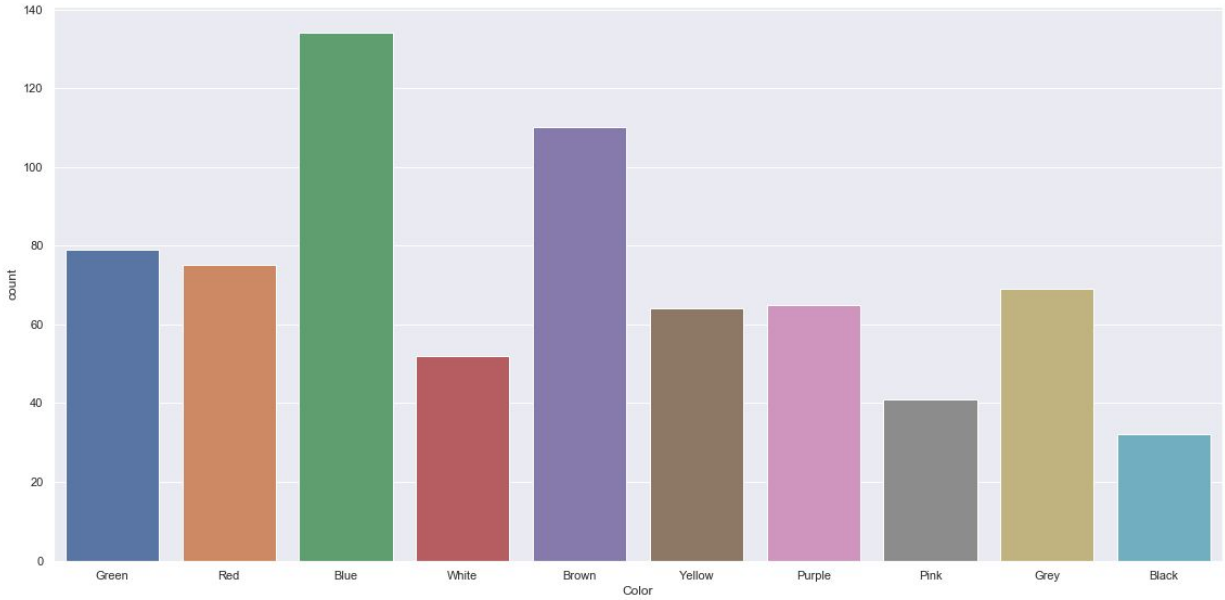


Gráfico 2: Gráfico de Distribuição de Cores

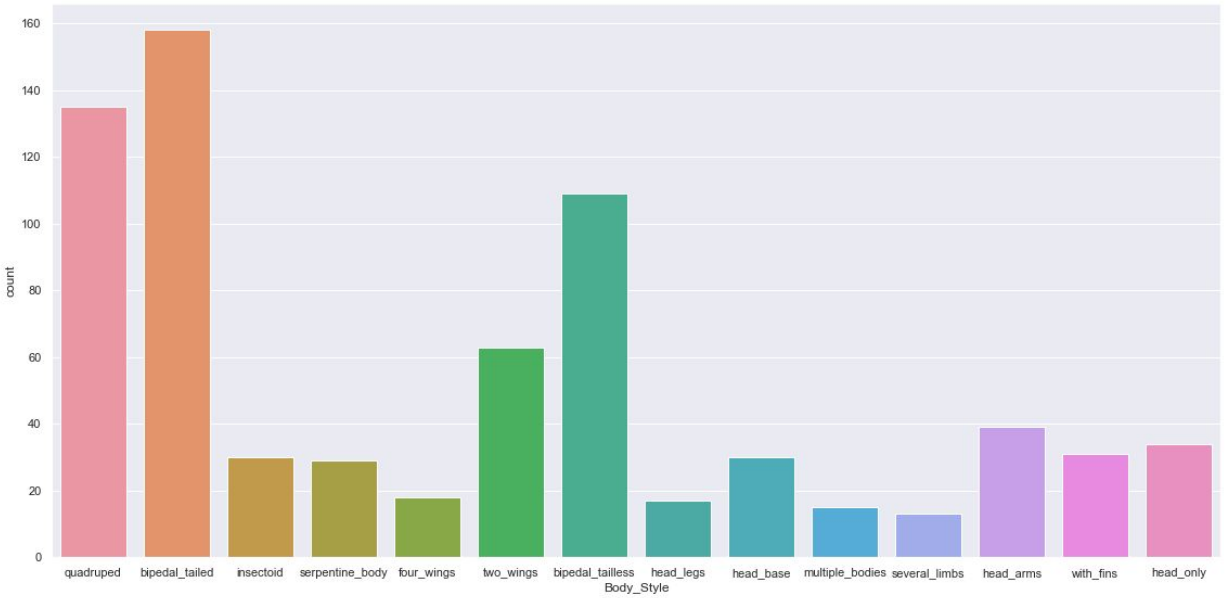


Gráfico 3 :Gráfico do Tipo físico dos pókemons

Em nosso trabalho utilizaremos, para o primeiro problema em que tentamos prever se um pokémon é lendário ou não, isLegendary como saída e como entrada as estatísticas do pokémon (HP, ataque, Sp_ataque,defesa, Sp_def, Velocidade) .No segundo problema , tentar prever o tipo do pokémon , iremos utilizar Type_1 como saída

e como entradas a cor , o tipo do corpo do pokémon , e o grupo do ovo. Para obtermos mais resultado iremos usar variar esse atributos no treinamento dos modelo.

Nossa classes para classificação serão o Boleano isLegendary para uma classificação binária (1 ou 0) e para o segundo problema teremos 18 classes que representam os tipos dos pokémons .

Existem alguns trabalhos similares, podemos citar,por exemplo, a classificação binária se um pokémon é lendário ou não baseado nos status. Utilizando o algoritmo de KNN foi obtida uma taxa de acerto de superior a 95%, usando o algoritmo de floresta aleatória foi obtida uma taxa de acerto de cerca de 97%.

3.Metodologia:

Toda nosso trabalho foi feito em um ambiente do Anaconda para python 3.7 . Utilizamos as seguintes bibliotecas :

1. Numpy : Computação de funções matemáticas.
2. Pandas : Utilização do Dataframe como estrutura para armazenar nosso dataset e aplicar funções em cima dele.
3. Matplotlib e Seaborn : Criação de Gráficos e Matrizes de correlação e confusão.
4. SkitLearn : Utilização do classificador DecisionTreeClassifier e Matrizes de confusão

Antes de aplicar os algoritmos geramos gráficos para verificarmos as distribuições de nossos dados , tanto categóricos como numéricos , e também geramos uma matriz de correlação para vermos se é válida nossa análise .

No primeiro problema explorado , por se tratar um problema mais simples , utilizamos 3 algoritmos de machine learning para classificação : Regressão Logística (Gradiente Descendente) , Árvore de decisão e KNN.Já no segundo utilizamos 5 modelos para classificação multiclasse : Regressão Logística (Com gradiente descendente e Softmax), Árvore de decisão, KNN, Discriminante Gaussiano e Naive Bayes Gaussiano.

Em ambos os experimentos no início de cada execução , separamos nossos dados em 2 conjuntos , de treinamento e de teste, temos a opção de normalizar os dados (algo que não foi feito no primeiro experimento pois as ordens de grandeza importam para nossa classificação), aplicamos o One Hot encoding somente na classificação multiclasse e por fim criamos um conjunto de validação .

Na execução de cada algoritmos realizamos o grid search para os hiperparâmetros necessários e escolhemos os melhores para utilizar na avaliação , são esses :

- Regressão Logística : Passo de treinamento (alpha)
- Árvore de Decisão : Critério para a criação de nós , a profundidade máxima da árvore e o mínimo de amostras por folhas
- KNN : o número de vizinhos (k) e a métrica de distância (euclidiana , manhattan e mahalanobis)
- Discriminante Gaussiano e Naive Bayes não utilizam hiperparâmetros portanto não foi necessária a utilização do Grid Search .

Para mensurar a acurácia dos modelos gerados utilizamos os erros médios quadrados como função de custo e medimos a taxa de erro no conjunto de teste. Também utilizamos matrizes de confusão para visualizar quão bem os algoritmos classificaram as suas amostras .

4.Experimentos:

I :Experimento 1 : Classificação Binária

Em nosso primeiro problema utilizamos toda a metodologia descrita no item 3 ,utilizamos 2 parâmetros para a divisão de treino e teste , uma separação de 80% de dados para treino e 20% para teste e outra de 60% de dados para treino e 40% para teste .A acurácia em ambos os casos foi satisfatória sendo maior do que 90%, no entanto um fato que dificulta a nossa análise é a proporção dentre pokémons lendários e não lendários visto que 90% dos pokemons não são.

Total	Name	HP	Attack	Defense	Sp_Atk	Sp_Def	Speed	isLegendary
600	Meloetta	100	77	77	128	128	90	False
600	Heatran	91	90	106	130	106	77	True
600	Darkrai	70	90	90	135	90	125	True
600	Dragonite	91	134	95	100	100	80	False
600	Hydreigon	92	105	90	125	90	98	False

Tabela 1: Exemplo dos dados utilizados

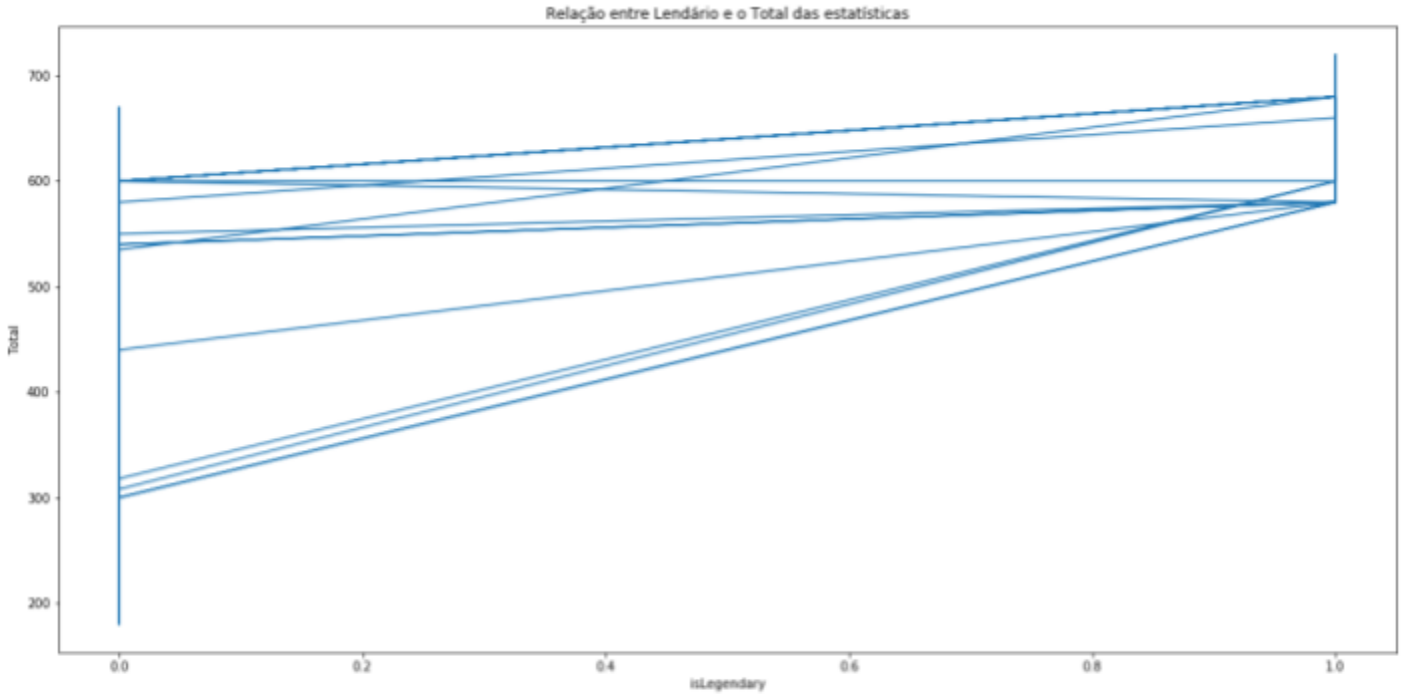
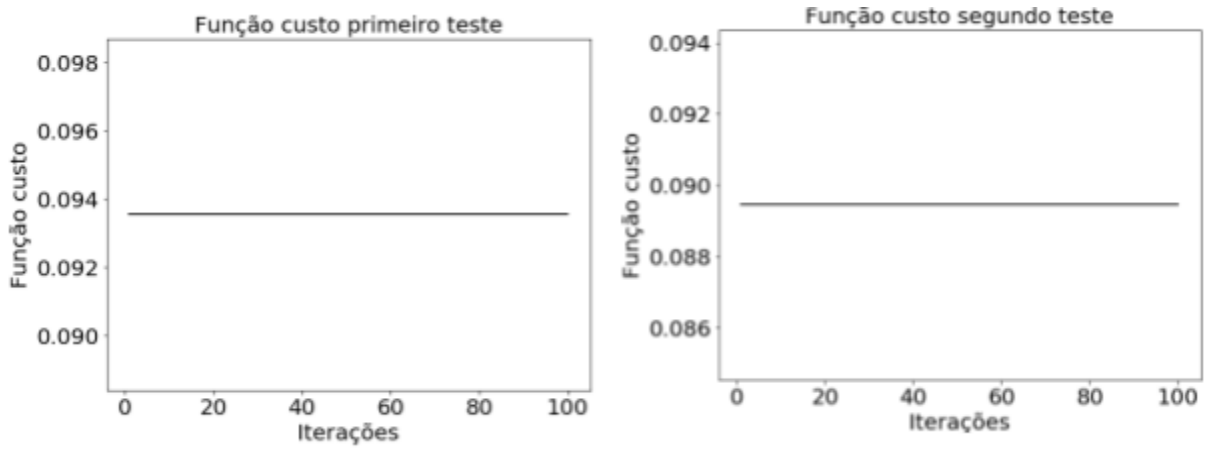


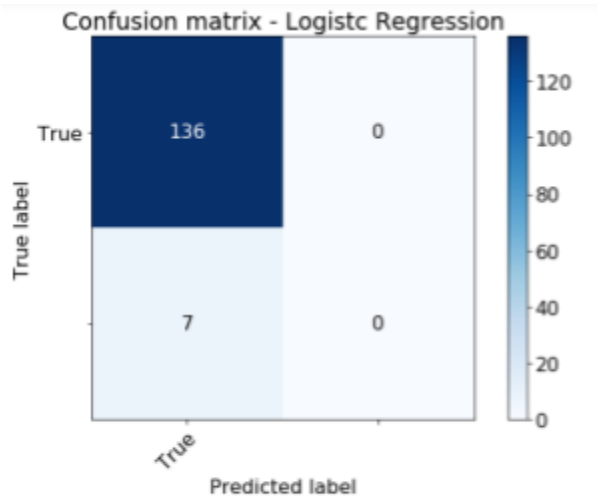
Gráfico 4: Relacionamento entre o Total da soma de todos os Status do Pokémon e se ele é ou não lendário

Regressão Logística : Número de épocas : 100 Melhor alpha em ambos os casos : 10^{-6} .

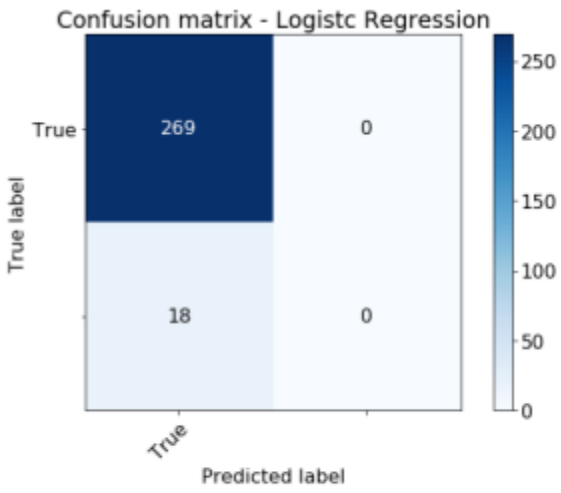
A função custo no primeiro e no segundo experimento se mantiveram constantes durante a classificação do algoritmo obtentendo 4.90% de erro no teste no primeiro caso e 6.27% no segundo caso . Resultados satisfatórios , mas que quando verificamos as matrizes de confusão notamos que o algoritmo simplesmente classificou todas as amostras como não lendários,fato que pode ser explicado pela proporção entre as 2 classes , por isso não escolhemos a regressão logística como melhor modelo para esse problema .



Função de custo ao longo de 100 iterações



Matriz de confusão primeira execução



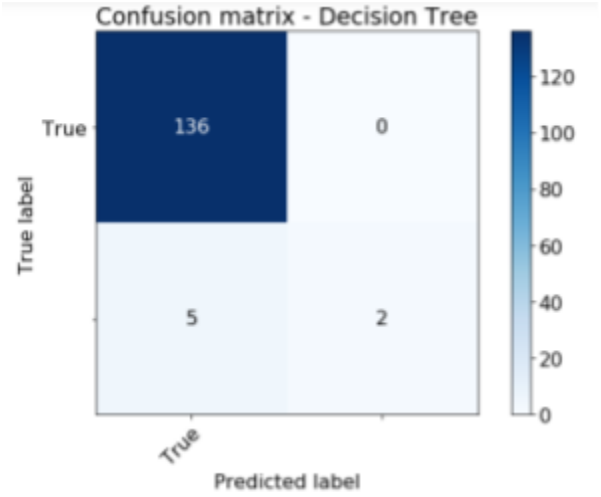
Matriz de confusão segunda execução

Árvore de decisão :

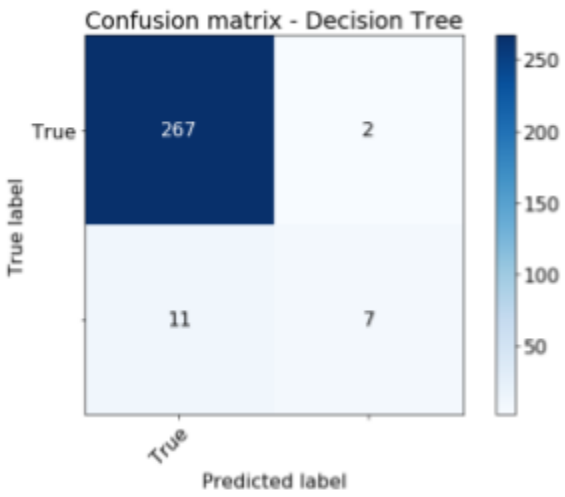
- 1° experimento : Melhor critério de similaridade : Entropia , profundidade da árvore : 3 e mínimo de amostras por folha :1.
- 2° experimento :Melhor critério de similaridade : Gini , profundidade da árvore : 2 e mínimo de amostras por folha :1

Em ambos os experimentos a árvore de decisão foi o modelo que apresentou a menor taxa de erros no teste , 3.50 % na primeira execução e 4.53%. Se mostrando o melhor classificador que a regressão Logística , já que diferente dela classificou corretamente lendários ao invés de só não

lendários como pode ficar evidente nas matrizes de confusão.



Matriz de confusão na primeira execução

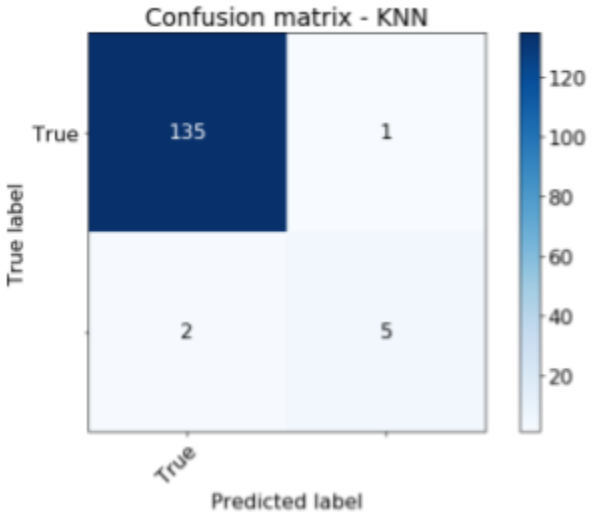


Matriz de confusão na segunda execução

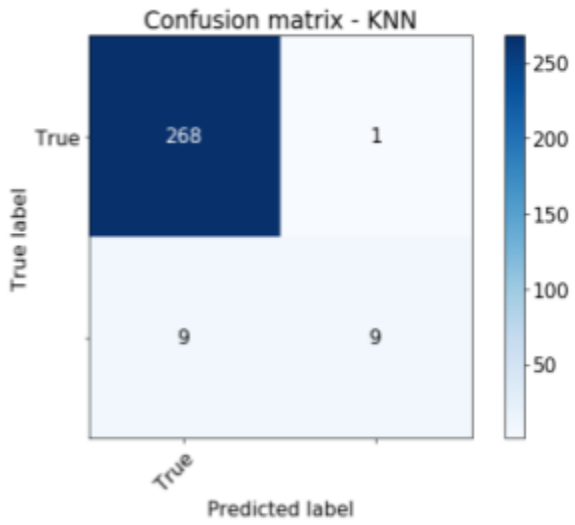
KNN :

- 1° experimento : Melhor número de vizinhos : 1 , Melhor métrica de distância : Euclidiana .
- 2° experimento : Melhor número de vizinhos : 3 e Melhor métrica de distância : Euclidiana .

  Na primeira execução o algoritmo KNN obteve uma Taxa de erro no teste de 2.10% e na segunda execução de 3.48% se mostrando o melhor classificador pois além de obter as melhores taxas também conseguiu classificar corretamente os dados em ambas as classes .



Matriz de confusão para o primeiro experimento



Matriz de confusão para o segundo experimento

II :Experimento 2 : Classificação Multiclasse

Para o nosso segundo problema realizamos 2 experimentos independentes , um somente com os dados das colunas Color e Body_Style para tentarmos prever o Type_1 e no outro experimento adicionando a coluna Egg_Group em nossas entradas . Utilizamos uma proporção de 80% para dados de treino e 20% para dados de teste em ambos .

Color	Name	Type_1	Body_Style	Egg_Group_1
Blue	Mantyke	Water	two_wings	Undiscovered
Blue	Quagsire	Water	bipedal_tailed	Water_1
Blue	Totodile	Water	bipedal_tailed	Monster
Blue	Chinchou	Water	with_fins	Water_2
Blue	Ducklett	Water	two_wings	Water_1

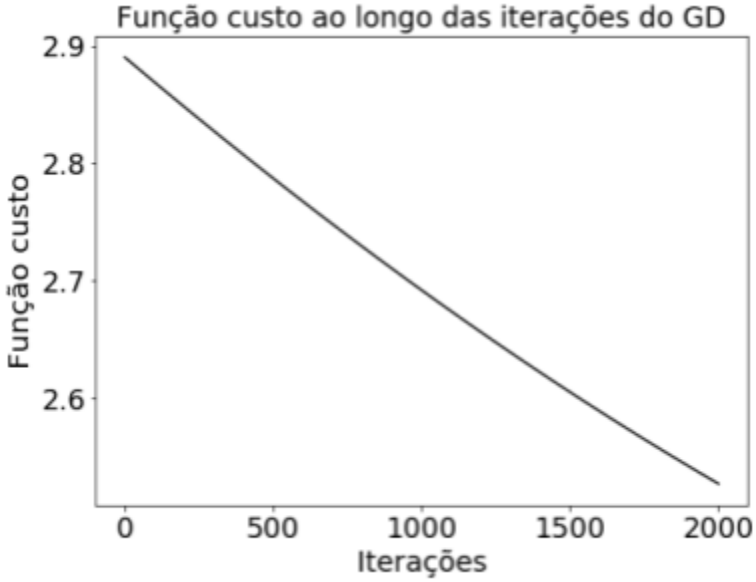
Tabela 2: Exemplo dos Dados utilizados na aprendizagem agrupados por Cor

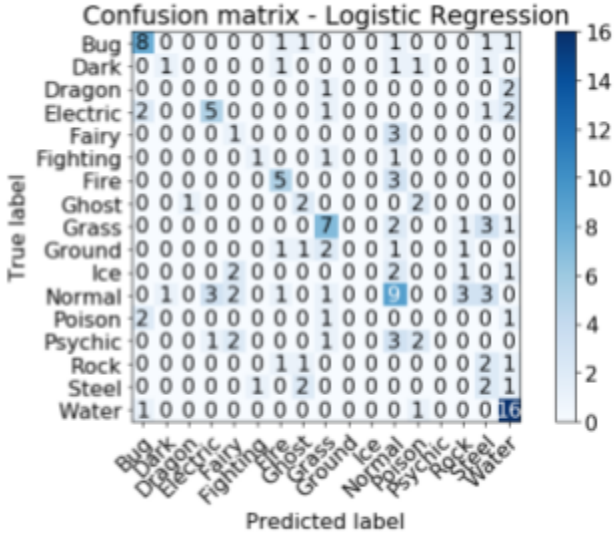
Type_1	Name	Color	Body_Style	Egg_Group_1
Dragon	Garchomp	Blue	bipedal_tailed	Monster
Dragon	Druddigon	Red	bipedal_tailed	Dragon
Dragon	Dragonair	Blue	serpentine_body	Water_1
Dragon	Goomy	Purple	serpentine_body	Dragon
Dragon	Goodra	Purple	bipedal_tailed	Dragon

Tabela 3 : Exemplo de dados utilizados na aprendizagem agrupados por Tipo do Pokémon

No primeiro experimento os resultados foram bem piores do que os esperados , nenhum dos modelos conseguiu uma taxa de acerto acima de 50% e conseguimos através das matrizes de confusão verificar os tipos melhores e piores classificados.

Regressão Logística : Melhor passo de aprendizagem 10^{-3}. Rodando em 2000 épocas, conseguimos uma função custo estritamente decrescente , empatando como melhor classificador tendo cerca de 60% de erro teste. É interessante verificar na matriz de confusão que a regressão acerta bem alguns tipos como: Water , Bug, Electric , Normal e Grass enquanto erra bastante

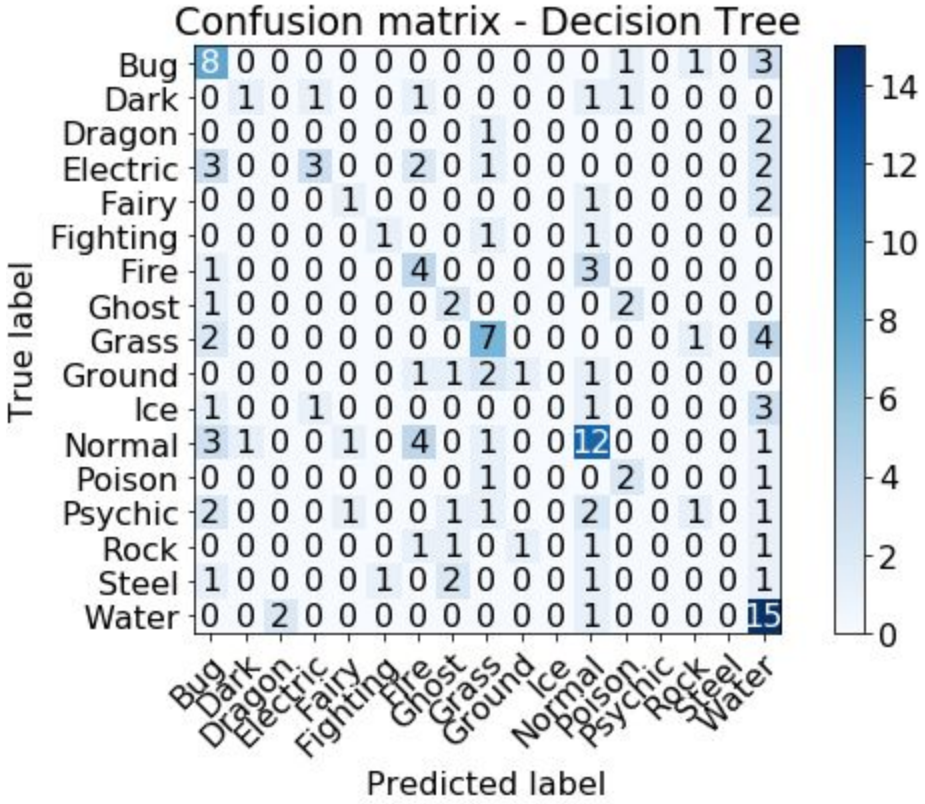




outros como : Ground e Psychic.

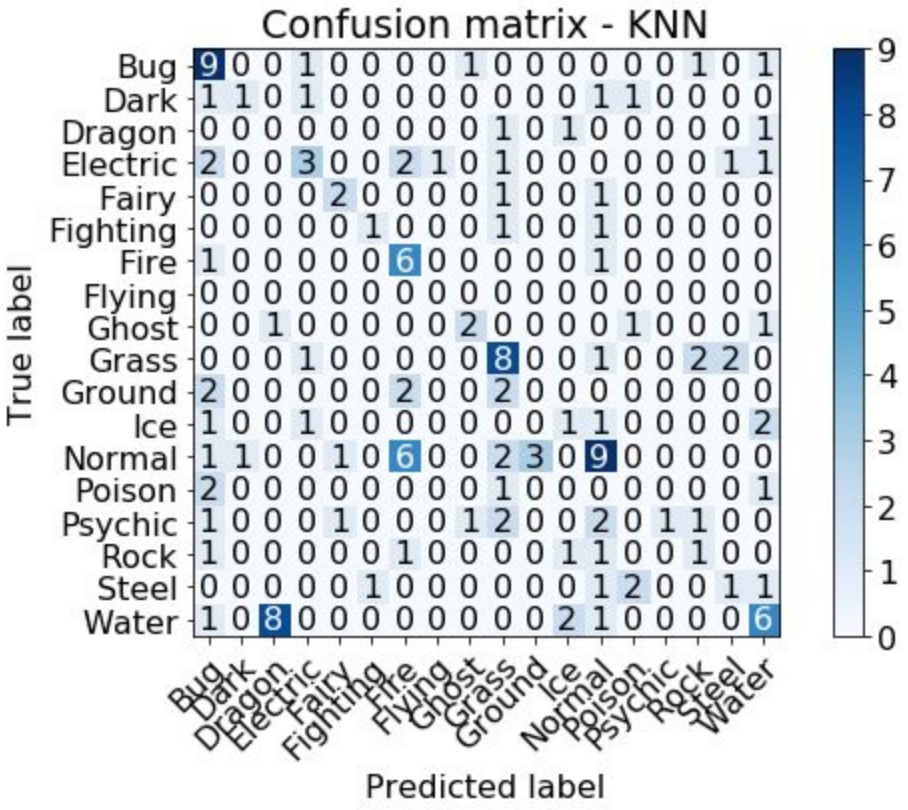
Árvore de Decisão : Melhor critério de dissimilaridade : entropia , Melhor Profundidade da árvore : 11 , Melhor mínimo de nós nas folhas :3

A árvore de decisão obteve resultados bem semelhantes a regressão logística , mesma taxa de erro no teste , de 60%, e ainda classificou corretamente os tipos de modo semelhante ao modelo anterior com uma pequena diferença de que acertou mais dados da classe Fire .

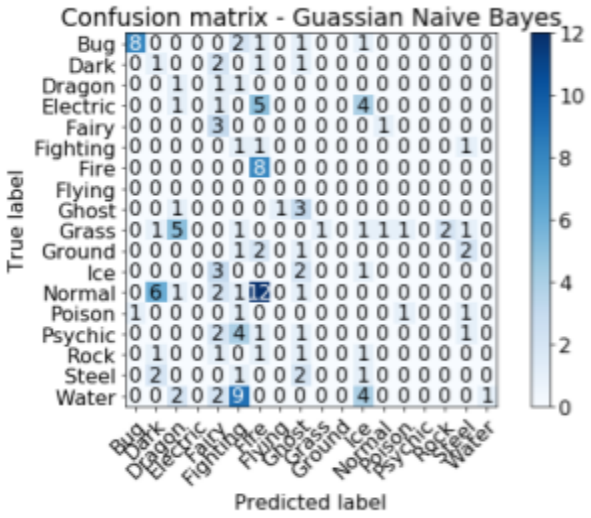


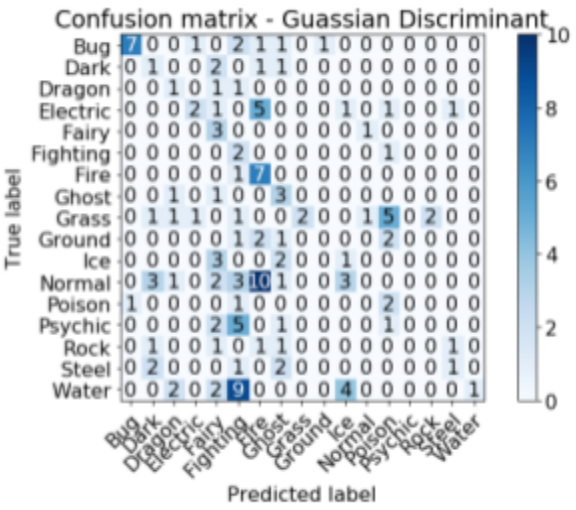
KNN : Melhor número de vizinhos : 7 e melhor métrica de distância : mahalanobis .

O KNN foi o classificador que obteve o segundo melhor desempenho obtendo uma taxa de erro no teste de 64% e acertando os mesmos tipos , no entanto errando mais a classe Water e acertando mais da classe Fire se comparado com os anteriores .



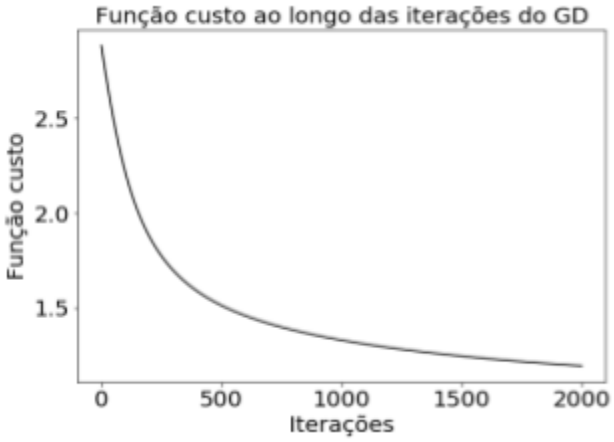
Naive Bayes e Análise de discriminante Gaussiano foram os 2 piores modelos obtendo respectivamente taxas de erro no teste de 79.72% e 76.92% , fato que demonstra que tentar aproximar a distribuição de nossos dados a uma gaussiana não é interessante, visto que a taxa de acerto é pífia .Na questão das classes mais acertadas , ambos os modelos continuam acertando bem as classes Bug e Fire , no entanto errando as classes Normal ,Grass e Water que os outros modelos classificam muito bem .

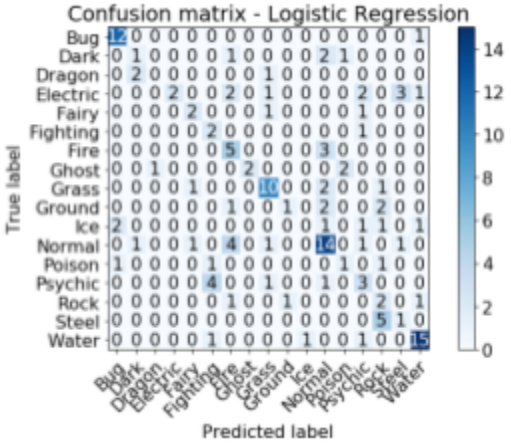




Agora para o segundo experimento adicionamos um novo atributo o Egg_Group_1 ,realizamos o processo descrito em nossa metodologia e a mesma proporção entre conjuntos de teste e validação. Esse atributo contribuiu consideravelmente com a classificação visto que reduziu em cerca de 10% a taxa de erro em todos o modelos, esse fato pode se consequência que o Egg_Group é diretamente relacionado com o tipo do pokémon , não necessariamente com suas características físicas .

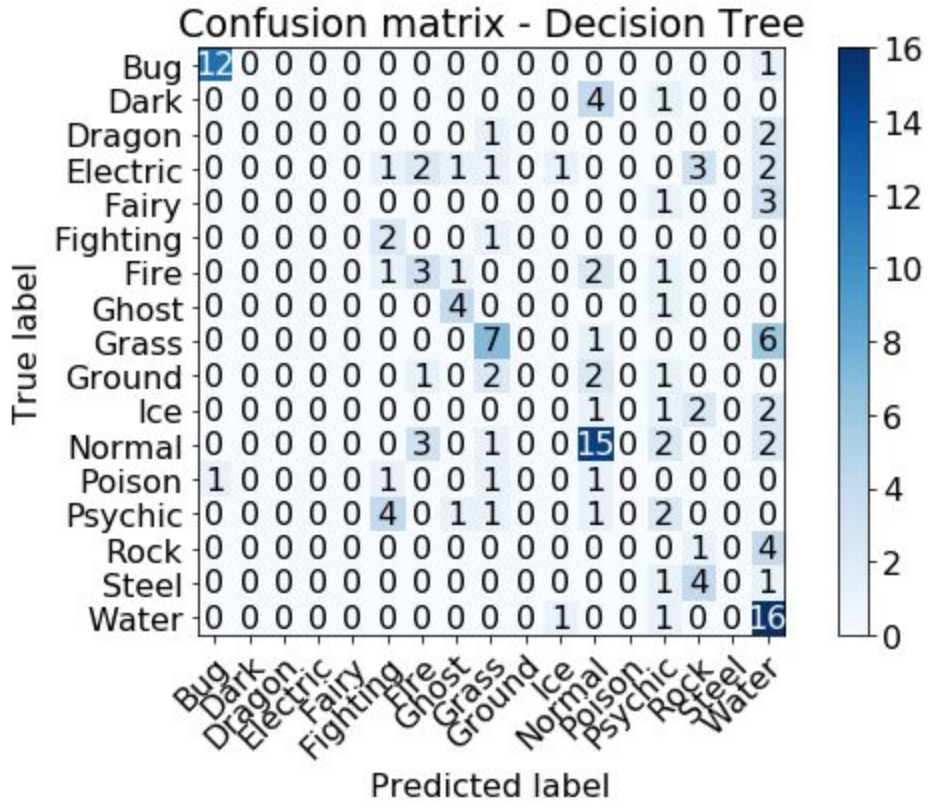
Regressão Logística : Melhor passo de aprendizagem $1.72*10^{-2}$. Rodando em 2000 épocas, conseguimos uma função custo que se estabiliza após cerca de 500 épocas , empatando como melhor classificador tendo cerca de 48% de erro teste. É interessante verificar na matriz de confusão que a regressão acerta bem algumas classes como: Water , Bug, Fire , Normal e Grass enquanto erra bastante as classes como Eletric e Psychic e Ice.





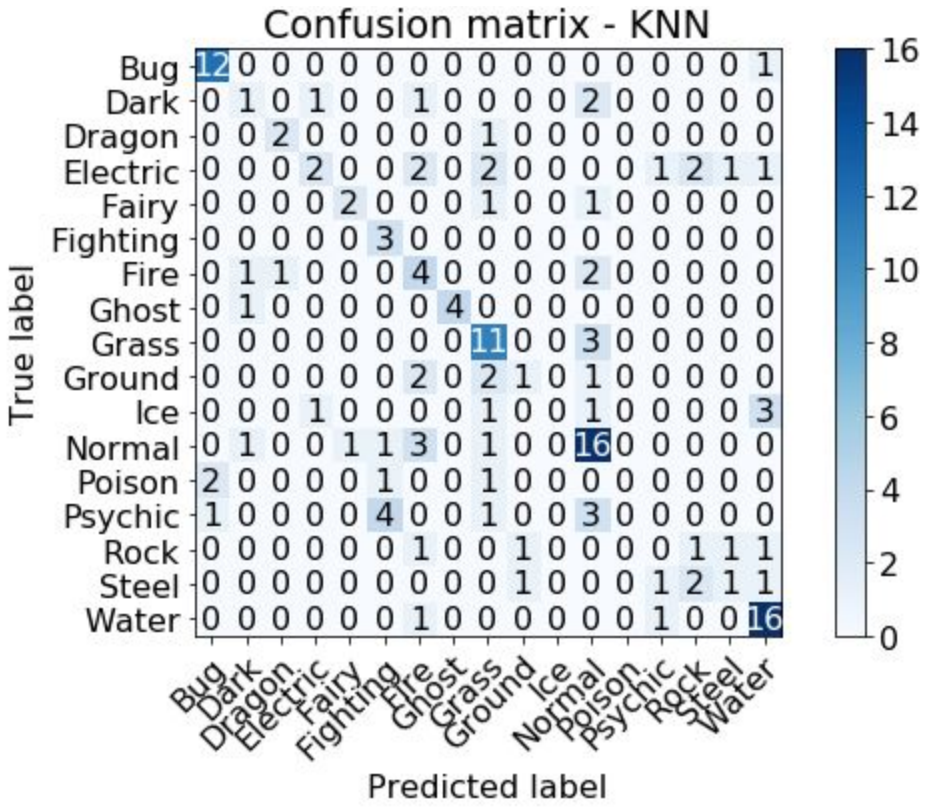
Árvore de Decisão : Melhor critério de dissimilaridade : entropia , Melhor Profundidade da árvore : 9 , Melhor mínimo de nós nas folhas :13

A árvore de decisão obteve resultados melhores do que no experimento 1 porém não melhorou o suficiente para se equiparar com os melhores modelos , ficando com uma taxa de erro de 56.64% e no critério dos tipos melhores avaliados temos os mesmo do experimento 1 porém com mais amostras sendo corretamente classificados . É importante perceber que o novo atributo adicionado reduziu a profundidade da árvore e aumentando o número de amostras por folha .

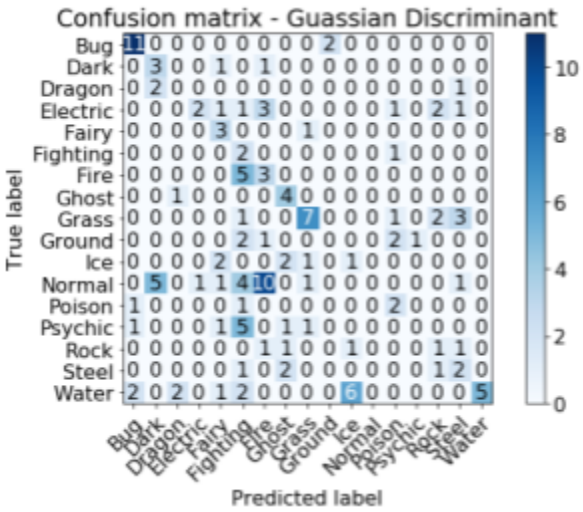


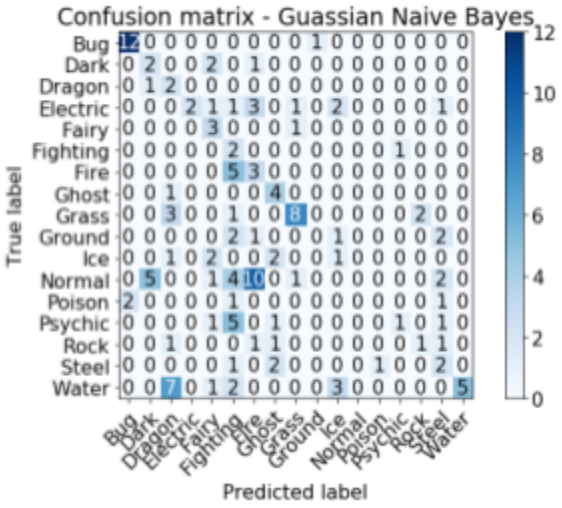
KNN: Melhor número de vizinhos : 13 e melhor métrica de distância : euclidiana .

O KNN foi o classificador que obteve o melhor desempenho dentre todos os classificadores empatando com a regressão logística , visto que a sua taxa de acerto foi de 46.85% e no critério dos tipos melhores classificados foram os mesmos dos anteriores , no entanto acertando mais amostras como pode ser conferido na matriz de confusão.



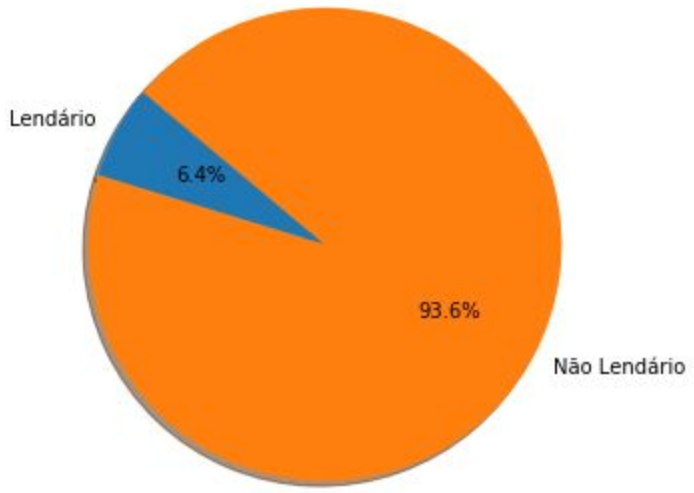
Naive Bayes e Análise de discriminante Gaussiano mesmo com a adição do novo atributo continuaram sendo os 2 piores modelos obtendo respectivamente taxas de erro no teste de 66.43% e 67.83% , fato que ,mais uma vez ,demonstra que tentar aproximar a distribuição de nossos dados a uma gaussiana não é interessante,mas nesse caso funciona melhor que no experimento 1 .Na questão das classes mais acertadas , ambos os modelos continuam acertando bem as classes Bug Grass e Fire , no entanto errando as classes Normal e Water que os outros modelos classificam muito bem .





5.Conclusão:

Após a execução do problema de classificação dentre lendários e não lendários vemos que é possível criar modelos que façam tal predição , no entanto uma das dificuldades enfrentadas é a quantidade de dados de pokémons lendários que são bem menores de que os de não lendários .



No caso das execuções do problema de classificação podemos tirar algumas conclusões : os modelos tendem a predizer corretamente as classes que possuem mais amostras no nosso dataset como pode ser observado nas classes Water e Normal e Bug são bem classificadas por nossos modelos , outro fato é que quando adicionamos o novo atributo Egg_Group_1 a acurácia subiu bastante visto que esse atributo está diretamente relacionado com nossas classes de saída . A utilização de classificadores estatísticos nesse problema não é interessante pois a aproximação dos dados para uma distribuição Gaussiana não funciona muito bem visto que são os modelos com as piores taxa de erros . O uso de classificadores para esse problema é aceitável já que obtemos uma acurácia boa cerca de 50% , no entanto de o dado não permanecer as classes dominantes geralmente ele será classificado erroneamente.

Se tivéssemos mais tempo poderíamos realizar outras análises dentro do universo de dados de Pokémon , como um sistema de recomendação de membros de times competitivos , esse que necessitaria uma coleta de novos dados mais relacionados a esse problema . Outra projeto que poderíamos realizar também seria expandir esse projeto utilizando dados na 7ª e 8ª gerações de Pokémons , assim aumentando a quantidade de dados .